Lista de Exercícios - Ponteiros e alocação dinâmica

BCC702 - Programação de Computadores II

2025.1

INSTRUÇÕES

- A atividade é individual.
- Implemente **TODOS** os exercícios.
- Desenvolver o código e realizar os testes em C++.
- Incluir comentários no código explicando as etapas. Comente trechos do código, não linha a linha!
- Envie suas soluções no Moodle. Para enviar, crie um arquivo compactado com todas as suas soluções e envie o arquivo compactado.
- Exercícios enviados fora do prazo serão penalizados com 10% da nota por dia de atraso.

Exercício 1

Para cada trecho de código abaixo, qual é o valor atribuído à variável r?

Trecho 1

```
int x = 5, y = 2;
int *z = &x;
int *w = new int;
*w = *z;
int r = x + y + *w + *z;
```

Trecho 2

```
bool b1 = true, b2 = false, b3 = true;
bool *c = &b2;
bool d = b3;
bool r = b1 and *c and d;
```

Trecho 3

```
float a = 2.3, *b = &a, c = 4.5;
float v[3] = {1.1, 2.2, 3.3};
float *x = new float[3];
float *y = v;
x[0] = a;
x[1] = *b;
x[2] = y[2] + c;
float r = x[0] + x[1] + x[2];
```

Exercício 2

Escreva uma função que receba um vetor de inteiros e seu tamanho, e conte quantos números pares e quantos ímpares existem nesse vetor. Os totais devem ser armazenados em duas variáveis passadas por ponteiro.

Sua função deve ser do tipo `void` e recebe, no total, quatro parâmetros: um ponteiro para o vetor, o tamanho do vetor e dois ponteiros onde devem ser armazenados o total de pares e ímpares.

Faça também uma função `main` que leia o tamanho e os elementos do vetor, chame a função e imprime o resultado. Aloque dinamicamente o vetor dentro da `main`.

```
Digite o tamanho do vetor: 6
Digite os elementos: 2 5 8 3 1 10

Quantidade de pares: 3
Quantidade de ímpares: 3
```

Exercício 3

Crie um registro para representar um `Ponto`, que contém os campos x e y, que representam suas coordenadas. Crie também uma função que recebe por parâmetro dois pontos, calcula e retorna a distância euclidiana entre eles.

Distância Euclidiana:

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Onde: p representa o **ponto 1** e q representa o **ponto 2**. Utilize as funções `double sqrt(double x)` e `double pow(double base, double expoente)` disponíveis na biblioteca `cmath` (`#include <cmath>`)

Na função main, crie um vetor de pontos alocado dinamicamente de tamanho N, informado pelo usuário, e leia as coordenadas de cada ponto. Use a função definida para calcular o tamanho do caminho definido pelos pontos, na ordem em que foram lidos.

O tamanho total do caminho é igual a soma das distâncias entre pontos consecutivos.

Exemplo 1

```
Informe o número de pontos: 4
Informe a coordenadas do ponto 1: 3 3
Informe a coordenadas do ponto 2: 1 1
Informe a coordenadas do ponto 3: -1 2
Informe a coordenadas do ponto 4: 0 3

Tamanho do caminho: 6.47871
```

Exercício 4

Escreva um programa para ler uma matriz de inteiros ($A_{m \times n}$), um valor digitado pelo usuário e verifique se esse valor encontra-se na matriz.

O programa deve ler os índices m, n, a matriz A e um valor a ser procurado. O programa deve imprimir se encontrou o valor na matriz ("Valor encontrado!") ou não ("Valor não encontrado!").

As matrizes devem ser armazenadas utilizando-se alocação dinâmica. **Dica:** lembre-se que uma matriz é um vetor de vetores. Assim, precisamos primeiro alocar um vetor de ponteiros para cada linha. Depois, precisamos alocar um novo vetor para cada linha. Ao desalocar sua matriz, você deve primeiro desalocar as linhas e *depois* desalocar o vetor de ponteiros. Sua variável para a matriz terá então o tipo `int**`, por exemplo: `int **matriz`.

Exemplo 1

```
Digite o valor de m e n: 3 2
Digite a matriz:
1 3
2 6
-5 0
Digite o valor a ser procurado: -5

Valor encontrado!
```

Exemplo 2

Digite o valor de m e n: 4 4
Digite a matriz:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Digite o valor a ser procurado: -5

Valor não encontrado!